Context-Specific Likelihood Weighting

Nitesh Kumar
Department of Computer Science
KU Leuven, Belgium

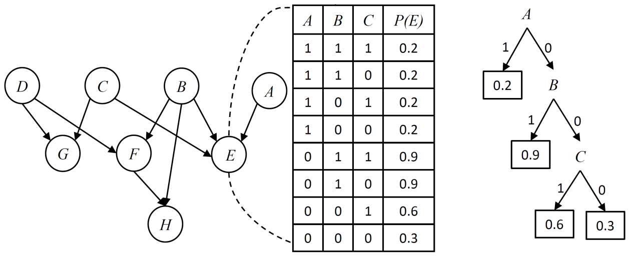
Ondřej Kuželka
Department of Computer Science
Czech Technical University in Prague, Czechia

Abstract

Sampling is a popular method for approximate inference when exact inference is impractical. Generally, sampling algorithms do not exploit context-specific independence (CSI) properties of probability distributions. We introduce context-specific likelihood weighting (CS-LW), a new sampling methodology, which besides exploiting the classical conditional independence properties, also exploits CSI properties. Unlike the standard likelihood weighting, CS-LW is based on partial assignments of random variables and requires fewer samples for convergence due to the sampling variance reduction. Furthermore, the speed of generating samples increases. Our novel notion of contextual assignments theoretically justifies CS-LW. We empirically show that CS-LW is competitive with state-of-the-art algorithms for approximate inference in the presence of a significant amount of CSIs.

1 Introduction

Exploiting independencies present in probability distributions is crucial for feasible probabilistic inference. Bayesian networks (BNs) qualitatively represent *conditional independencies* (CIs) over random variables, which allow inference algorithms to exploit them. In many applications, however, exact inference quickly becomes infeasible. The use of stochastic sampling for approximate inference is common in such applications. Sampling algorithms are simple yet powerful tools for inference. They can be applied to arbitrary complex distributions, which is not true for exact inference algorithms. The design of efficient sampling algorithms for BNs has received much attention in the past. Unfortunately, BNs can not represent certain independencies qualitatively: independencies that hold only in certain contexts (Boutilier et al., 1996). These independencies are called *context-specific independencies* (CSIs). To illustrate them, consider a BN in Figure 1, where a tree-structure is present in the *conditional probability distribution* (CPD) of a random variable E. If one observes the CPD carefully, they can conclude that $P(E \mid A = 1, B, C) = P(E \mid A = 1)$, that is, $P(E \mid A = 1, B, C)$ is same for all values of B and C. The variable E is said to be independent of variables $\{B, C\}$ in the context $A = 1$. These independencies may have global implications, for instance, $E \perp B, C \mid A = 1$ implies $E \perp D \mid H, A = 1$. Sampling algorithms generally do not exploit CSIs arising due to structures within CPDs.



A	B	C	P(E)
1	1	1	0.2
1	1	0	0.2
1	0	1	0.2
1	0	0	0.2
0	1	1	0.9
0	1	0	0.9
0	0	1	0.6
0	0	0	0.3

Figure 1: Context-Specific Independence

One might think that structures in CPDs are accidental. It turns out, however, that such structures are common in many real-world settings. For example, consider a scenario (Koller and Friedman, 2009) where a symptom, *fever*, depends on 10 diseases. It would be impractical for medical experts to answer $1,024$ questions of the format: "What is the probability of high fever when the patient has disease A does not have disease B ...?" It might be the case that, if patients suffer from disease A, then they are certain to have a high fever, and our knowledge of their suffering from other diseases does not matter. One might argue, what if we automatically learn BNs from data? In this case, however, a huge amount of data would be needed to learn the parameters that are exponential in the number of parents required to describe the tabular-CPD. The tree-CPDs that require much fewer parameters are a more efficient way of learning BNs automatically from data (Chickering et al., 1997; Friedman, 1998; Breese et al., 1998). Moreover, the structures naturally arise due to *if-else conditions* in programs written in

Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130. Copyright 2021 by the author(s).

probabilistic programming languages (PPLs).

There are exact inference algorithms that exploit CSIs, and thus, form state-of-the-art algorithms for exact inference (Friedman and Van den Broeck, 2018). These algorithms are based on the knowledge compilation technique (Darwiche, 2003) that uses logical reasoning to naturally exploit CSIs. An obvious question, then, is: *how to design a sampling algorithm that naturally exploits CSIs, along with CIs?* It is widely believed that CSI properties in distributions are difficult to harness for approximate inference (Friedman and Van den Broeck, 2018). In this paper, we answer this difficult question by developing a sampling algorithm that can harness both CI and CSI properties.

To realize this, we adopt *likelihood weighting* (LW, Shachter and Peot, 1990; Fung and Chang, 1990), a sampling algorithm for BNs; and extend it to a rule-based representation of distributions since rules are known to represent the structures qualitatively (Poole, 1997). We call the resulting algorithm *context-specific likelihood weighting* (CS-LW) and provide its open-source implementation[1]. Additionally, we present a novel notion of *contextual assignments* that provides a theoretical framework for exploiting CSIs. Taking advantage of the better representation of structures via rules, CS-LW assigns only a subset of variables required for computing conditional query leading to i) faster convergence, ii) faster speed of generating samples. This contrasts with many modern sampling algorithms such as collapsed sampling, which speed up convergence by sampling only a subset of variables but at the cost of much reduced speed of generating samples. We empirically demonstrate that CS-LW is competitive with state of the art.

2 Background

We denote random variables with uppercase letters (A) and their assignments with lowercase letters (a). Bold letters denote sets ($\mathbf{A}$) and their assignments ($\mathbf{a}$). Parents of the variable A are denoted with $\mathbf{Pa}(A)$ and their assignments with $\mathbf{pa}(A)$. In a probability distribution $P(\mathbf{E}, \mathbf{X}, \mathbf{Z})$ specified by a Bayesian network $\mathcal{B}$, $\mathbf{E}$ denotes a set of observed variables, $\mathbf{X}$ a set of unobserved query variables and $\mathbf{Z}$ a set of unobserved variable other than query variables. The expected value of A relative to a distribution Q is denoted by $\mathbb{E}_Q[A]$. Next, we briefly introduce LW, one of the most popular approximate inference algorithms for BNs.

2.1 Likelihood Weighting

A typical query to a probability distribution $P(\mathbf{E}, \mathbf{X}, \mathbf{Z})$ is to compute $P(\mathbf{x}_q \mid \mathbf{e})$, that is, the probability of $\mathbf{X}$ being assigned $\mathbf{x}_q$ given that $\mathbf{E}$ is assigned $\mathbf{e}$. Following Bayes's rule, we have:

$$P(\mathbf{x}_q \mid \mathbf{e}) = \frac{P(\mathbf{x}_q, \mathbf{e})}{P(\mathbf{e})} = \frac{\sum_{\mathbf{x},\mathbf{z}} P(\mathbf{x}, \mathbf{z}, \mathbf{e}) f(\mathbf{x})}{\sum_{\mathbf{x},\mathbf{z}} P(\mathbf{x}, \mathbf{z}, \mathbf{e})} = \mu,$$

where $f(\mathbf{x})$ is an indicator function $\mathbb{1}\{\mathbf{x} = \mathbf{x}_q\}$, which takes value 1 when $\mathbf{x} = \mathbf{x}_q$, and 0 otherwise. We can estimate μ using LW if we specify P using a Bayesian network $\mathcal{B}$. LW belongs to a family of importance sampling schemes that are based on the observation,

$$\mu = \frac{\sum_{\mathbf{x},\mathbf{z}} Q(\mathbf{x}, \mathbf{z}, \mathbf{e}) f(\mathbf{x}) (P(\mathbf{x}, \mathbf{z}, \mathbf{e}) / Q(\mathbf{x}, \mathbf{z}, \mathbf{e}))}{\sum_{\mathbf{x},\mathbf{z}} Q(\mathbf{x}, \mathbf{z}, \mathbf{e}) (P(\mathbf{x}, \mathbf{z}, \mathbf{e}) / Q(\mathbf{x}, \mathbf{z}, \mathbf{e}))}, \quad (1)$$

where Q is a *proposal distribution* such that $Q > 0$ whenever $P > 0$. The distribution Q is different from P and is used to draw independent samples. Generally, Q is selected such that the samples can be drawn easily. In the case of LW, to draw a sample, variables $X_i \in \mathbf{X} \cup \mathbf{Z}$ are assigned values drawn from $P(X_i \mid \mathbf{pa}(X_i))$ and variables in $\mathbf{E}$ are assigned their observed values. These variables are assigned in a topological ordering relative to the graph structure of $\mathcal{B}$. Thus, the proposal distribution in the case of LW can be described as follows:

$$Q(\mathbf{X}, \mathbf{Z}, \mathbf{E}) = \prod_{X_i \in \mathbf{X} \cup \mathbf{Z}} P(X_i \mid \mathbf{Pa}(X_i)) \mid_{\mathbf{E}=\mathbf{e}}.$$

Consequently, it is easy to compute the *likelihood ratio* $P(\mathbf{x}, \mathbf{z}, \mathbf{e}) / Q(\mathbf{x}, \mathbf{z}, \mathbf{e})$ in Equation 1. All factors in the numerator and denominator of the fraction cancel out except for $P(x_i \mid \mathbf{pa}(X_i))$ where $x_i \in \mathbf{e}$. Thus,

$$\frac{P(\mathbf{X}, \mathbf{Z}, \mathbf{e})}{Q(\mathbf{X}, \mathbf{Z}, \mathbf{e})} = \prod_{x_i \in \mathbf{e}} P(x_i \mid \mathbf{Pa}(X_i)) = \prod_{x_i \in \mathbf{e}} W_{x_i} = W_{\mathbf{e}},$$

where W_{x_i}, which is also a random variable, is the *weight* of evidence x_i. The *likelihood ratio* $W_{\mathbf{e}}$ is the product of all of these weights, and thus, it is also a random variable. Given M independent weighted samples from Q, we can estimate:

$$\hat{\mu} = \frac{\sum_{m=1}^{M} f(\mathbf{x}[m]) w_{\mathbf{e}}[m]}{\sum_{m=1}^{M} w_{\mathbf{e}}[m]}. \quad (2)$$

2.2 Context-Specific Independence

Next, we formally define the independencies that arise due to the structures within CPDs.

Definition 1. *Let P be a probability distribution over variables $\mathbf{U}$, and let $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ be disjoint subsets of $\mathbf{U}$. The variables $\mathbf{A}$ and $\mathbf{B}$ are independent given $\mathbf{D}$ and context $\mathbf{c}$ if $P(\mathbf{A} \mid \mathbf{B}, \mathbf{D}, \mathbf{c}) = P(\mathbf{A} \mid \mathbf{D}, \mathbf{c})$ whenever $P(\mathbf{B}, \mathbf{D}, \mathbf{c}) > 0$. This is denoted by $\mathbf{A} \perp \mathbf{B} \mid \mathbf{D}, \mathbf{c}$. If $\mathbf{D}$ is empty then $\mathbf{A}$ and $\mathbf{B}$ are independent given context $\mathbf{c}$, denoted by $\mathbf{A} \perp \mathbf{B} \mid \mathbf{c}$.*

[1] The code is available here: `https://github.com/niteshroyal/CS-LW.git`

Independence statements of the above form are called *context-specific independencies* (CSIs). When $\mathbf{A}$ is independent of $\mathbf{B}$ given all possible assignments to $\mathbf{C}$ then we have: $\mathbf{A} \perp \mathbf{B} \mid \mathbf{C}$. The independence statements of this form are generally referred to as *conditional independencies* (CIs). Thus, CSI is a more fine-grained notion than CI. The graphical structure in $\mathcal{B}$ can only represent CIs. Any CI can be verified in linear time in the size of the graph. However, verifying any arbitrary CSI has been recently shown to be coNP-hard (Corander et al., 2019).

2.3 Context-Specific CPDs

A natural representation of the structures in a CPD is via a *tree-CPD*, as illustrated in Figure 1. For all assignments to the parents of a variable A, a unique leaf in the tree specifies a (conditional) distribution over A. The path to each leaf dictates the contexts, i.e., *partially assigned parents*, given which this distribution is used. It is easier to reason using tree-CPDs if we break them into finer-grained elements. A finer-grained representation of structured CPDs is via rules (Poole, 1997; Koller and Friedman, 2009), where each path from the root to a leaf in each tree-CPD maps to a rule. For our purposes, we will use a simple rule-based representation language, which can be seen as a restricted fragment of *Distributional Clauses* (DC, Gutmann et al., 2011).

Example 1. A set of rules for the tree-CPD in Figure 1:

```
e ~ bernoulli(0.2) ← a=1.
e ~ bernoulli(0.9) ← a=0 ∧ b=1.
e ~ bernoulli(0.6) ← a=0 ∧ b=0 ∧ c=1.
e ~ bernoulli(0.3) ← a=0 ∧ b=0 ∧ c=0.
```

We can also represent structures in CPDs of discrete-continuous distributions using this form of rules like this:

Example 2. Consider a machine that breaks down if the cooling of the machine is not working or the ambient temperature is too high. The following set of rules specifies a distribution over `cool`, `t`(temperature) and `broken`, where a CSI is implied: `broken` is independent of `cool` in a context `t>30`.

```
cool ~ bernoulli(0.1).
t ~ gaussian(25,2.2).
broken ~ bernoulli(0.9) ← t>30.
broken ~ bernoulli(0.6) ← t=<30 ∧ cool=0.
broken ~ bernoulli(0.1) ← t=<30 ∧ cool=1.
```

Intuitively, the *head* of a rule (`h` $\sim \mathcal{D} \leftarrow$ `b1` $\wedge \cdots \wedge$ `bn`) defines a random variable `h`, distributed according to a distribution $\mathcal{D}$, whenever all atoms `bi` in the *body* (an assignment of some parents of the variable) of the rule are true, that is: $p(\texttt{h} \mid \texttt{b1}, \ldots, \texttt{bn}) = \mathcal{D}$. Since we study tree-CPDs, we focus on *mutually exclusive and exhaustive* rules; that is, only one rule for the variable `h` can *fire* (each atom in the body of the rule is true) at a time. A set of rules forms a *program*, which we call the DC($\mathcal{B}$) program.

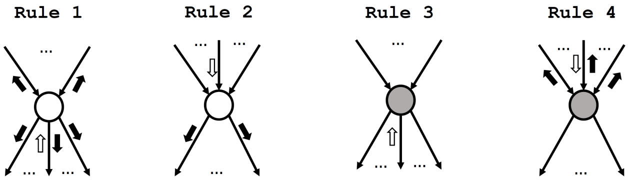


Figure 2: The four rules of Bayes-ball algorithm that decide next visits (indicated using ⬆) based on the direction of the current visit (indicated using ⇧) and the type of variable. To distinguish observed variables from unobserved variables, the former type of variables are shaded.

Definition 2. *Let $\mathcal{B}$ be a Bayesian network with tree-CPDs specifying a distribution P. Let $\mathbb{P}$ be a set of rules such that each path from the root to a leaf of each tree-CPD corresponds to a rule in $\mathbb{P}$. Then $\mathbb{P}$ specifies the same distribution P, and $\mathbb{P}$ will be called DC($\mathcal{B}$) program.*

3 Exploiting Conditional Independencies

In this section, we will ignore the structures within CPDs and only exploit the graphical structure of BNs. The approach presented in this section forms the basis of our discussion on CS-LW, where we will also exploit CPDs' structure.

In Section 2.1, we used all variables to estimate μ. However, due to CIs, observed states and CPDs of only some variables might be required for computing μ. These variables are called *requisite variables*. To get a better estimate of μ, it is recommended to use only these variables. The standard approach is to first apply the Bayes-ball algorithm (Shacter, 1998) over the graph structure in $\mathcal{B}$ to obtain a sub-network of requisite variables, then simulate the sub-network to obtain the weighted samples. An alternative approach that we present next is to use Bayes-ball to simulate the original network $\mathcal{B}$ and focus on only requisite variables to obtain the weighted samples.

To obtain the samples, we need to traverse the graph structure of the Bayesian network $\mathcal{B}$ in a topological ordering. The Bayes-ball algorithm, which is linear in the graph's size, can be used for it. The advantage of using Bayes-ball is that it also detects CIs; thus, it traverses only a sub-graph that depends on the query and evidence. We can also keep assigning unobserved variables, and weighting observed variables along with traversing the graph. In this way, we assign/weigh only requisite variables. The Bayes-ball algorithm uses four rules to traverse the graph (when deterministic variables are absent in $\mathcal{B}$), and marks variables to avoid repeating the same action. These rules are illustrated in Figure 2. Next, we discuss these rules and also indicate how to assign/weigh variables, resulting in a new algorithm called *Bayes-ball simulation of BNs*. Starting with all query variables scheduled to be visited as if from one of their children, we apply the following rules until no more variables can be visited:

1. When the visit of an unobserved variable $U \in \mathbf{X} \cup \mathbf{Z}$ is from a child, and U is not marked on top, then do these in the order: i) Mark U on top; ii) Visit all its parents; iii) Sample a value y from $P(U \mid \mathbf{pa}(U))$ and assign y to U; iv) If U is not marked on bottom, then mark U on bottom and visit all its children.

2. When the visit of an unobserved variable is from a parent, and the variable is not marked on bottom, then mark the variable on bottom and visit all its children.

3. When the visit of an observed variable is from a child, then do nothing.

4. When the visit of an observed variable $E \in \mathbf{E}$ is from a parent, and E is not marked on top, then do these in the order: i) Mark E on top; ii) Visit all its parents; iii) Let e be a observed value of E and let w be a probability at e according to $P(E \mid \mathbf{pa}(E))$, then the weight of E is w.

The above rules define an order for visiting parents and children so that variables are assigned/weighted in a topological ordering. Indeed we can define the order since the original rules for Bayes-ball do not prescribe any order. The marks record important information; consequently, we show the following. The proofs for all the results are in the supplementary material.

Lemma 1. *Let $\mathbf{E}_\star \subseteq \mathbf{E}$ be marked on top, $\mathbf{E}_{\smwhitestar} \subseteq \mathbf{E}$ be visited but not marked on top, and $\mathbf{Z}_\star \subseteq \mathbf{Z}$ be marked on top. Then the query μ can be computed as follows,*

$$\mu = \frac{\sum_{\mathbf{x},\mathbf{z}_\star} P(\mathbf{x}, \mathbf{z}_\star, \mathbf{e}_\star \mid \mathbf{e}_{\smwhitestar}) f(\mathbf{x})}{\sum_{\mathbf{x},\mathbf{z}_\star} P(\mathbf{x}, \mathbf{z}_\star, \mathbf{e}_\star \mid \mathbf{e}_{\smwhitestar})} \tag{3}$$

Now, since $\mathbf{X}, \mathbf{Z}_\star, \mathbf{E}_\star, \mathbf{E}_{\smwhitestar}$ are variables of $\mathcal{B}$ and they form a sub-network $\mathcal{B}_\star$ such that $\mathbf{E}_{\smwhitestar}$ do not have any parent, we can write,

$$P(\mathbf{x}, \mathbf{z}_\star, \mathbf{e}_\star \mid \mathbf{e}_{\smwhitestar}) = \prod_{u_i \in \mathbf{x} \cup \mathbf{z}_\star \cup \mathbf{e}_\star} P(u_i \mid \mathbf{pa}(U_i))$$

such that $\forall p \in \mathbf{pa}(U_i) : p \in \mathbf{x} \cup \mathbf{z}_\star \cup \mathbf{e}_\star \cup \mathbf{e}_{\smwhitestar}$. This means CPDs of some observed variables are not required for computing μ. Now we define these variables.

Definition 3. *The observed variables whose observed states and CPDs might be required to compute μ will be called diagnostic evidence.*

Definition 4. *The observed variables whose only observed states might be required to compute μ will be called predictive evidence.*

Diagnostic evidence (denoted by $\mathbf{e}_\star$) is marked on top, while predictive evidence (denoted by $\mathbf{e}_{\smwhitestar}$) is visited but not marked on top. The variables $\mathbf{X}$, $\mathbf{Z}_\star$, $\mathbf{E}_\star$, $\mathbf{E}_{\smwhitestar}$ will be called requisite variables. Now, we can sample from a factor $Q_\star$ of Q such that,

$$Q_\star(\mathbf{X}, \mathbf{Z}_\star, \mathbf{E}_\star \mid \mathbf{E}_{\smwhitestar}) = \prod_{X_i \in \mathbf{X} \cup \mathbf{Z}_\star} P(X_i \mid \mathbf{Pa}(X_i)) \,|_{\mathbf{E}_\star = \mathbf{e}_\star} \tag{4}$$

When we use Bayes-ball, precisely this factor is considered for sampling. Starting by first setting $\mathbf{E}_{\smwhitestar}$ their observed values, $\mathbf{X} \cup \mathbf{Z}_\star$ is assigned and $\mathbf{e}_\star$ is weighted in the topological ordering. Given M weighted samples $\mathcal{D}_\star = \langle \mathbf{x}[1], w_{\mathbf{e}_\star}[1] \rangle, \ldots, \langle \mathbf{x}[M], w_{\mathbf{e}_\star}[M] \rangle$ from $Q_\star$, we can estimate:

$$\tilde{\mu} = \frac{\sum_{m=1}^{M} f(\mathbf{x}[m]) w_{\mathbf{e}_\star}[m]}{\sum_{m=1}^{M} w_{\mathbf{e}_\star}[m]}. \tag{5}$$

In this way, we sample from a lower-dimensional space; thus, the new estimator $\tilde{\mu}$ has a lower variance compared to $\hat{\mu}$ due to the Rao-Blackwell theorem. Consequently, fewer samples are needed to achieve the same accuracy. Hence, we exploit CIs using the graphical structure in $\mathcal{B}$ for improved inference.

4 Exploiting CSIs

Now, we will exploit the graphical structure as well as structures within CPDs. This section is divided into two parts. The first part presents a novel notion of contextual assignments that forms a theoretical framework for exploiting CSIs. It provides an insight into the computation of μ using partial assignments of requisite variables. We will show that CSIs allow for breaking the main problem of computing μ into several sub-problems that can be solved independently. The second part presents CS-LW based on the notion introduced in the first part, where we will exploit the structure of rules in the program to sample variables given the states of only some of their requisite ancestors. This contrasts with our discussion till now for BNs where knowledge of all such ancestors' state is required.

4.1 Notion of Contextual Assignments

We will consider the variables $\mathbf{X}$, $\mathbf{Z}_\star$, $\mathbf{E}_\star$, $\mathbf{E}_{\smwhitestar}$ requisite for computing the query μ to the distribution P and the sub-network $\mathcal{B}_\star$ formed by these variables. We start by defining the partial assignments that we will use to compute μ at the end of this section.

Definition 5. *Let $\mathbf{Z}_\dagger \subseteq \mathbf{Z}_\star$ and $\mathbf{e}_\dagger \subseteq \mathbf{e}_\star$. Denote $\mathbf{Z}_\star \setminus \mathbf{Z}_\dagger$ by $\mathbf{Z}_\ddagger$, and $\mathbf{e}_\star \setminus \mathbf{e}_\dagger$ by $\mathbf{e}_\ddagger$. A partial assignment $\mathbf{x}$, $\mathbf{z}_\dagger$, $\mathbf{Z}_\ddagger$, $\mathbf{e}_\dagger$, $\mathbf{e}_\ddagger$ will be called contextual assignment if due to CSIs in P,*

$$\prod_{u_i \in \mathbf{x} \cup \mathbf{z}_\dagger \cup \mathbf{e}_\dagger} P(u_i \mid \mathbf{pa}(U_i)) = \prod_{u_i \in \mathbf{x} \cup \mathbf{z}_\dagger \cup \mathbf{e}_\dagger} P(u_i \mid \mathbf{ppa}(U_i))$$

where $\mathbf{ppa}(U_i) \subseteq \mathbf{pa}(U_i)$ is a set of partially assigned parents of U_i such that $\mathbf{Z}_\ddagger \cap \mathbf{Ppa}(U_i) = \varnothing$.

Example 3. *Consider the network of Figure 1, and assume that our diagnostic evidence is $\{F = 1, G = 0, H = 1\}$, predictive evidence is $\{D = 1\}$, and query is $\{E = 0\}$. From the CPD's structure, we have: $P(E = 0 \mid A = 1, B, C) = P(E = 0 \mid A = 1)$; consequently, a contextual assignment is $\mathbf{x} = \{E = 0\}, \mathbf{z}_\dagger = \{A = 1\}, \mathbf{e}_\dagger = \{\}, \mathbf{Z}_\ddagger = \{B, C\}, \mathbf{e}_\ddagger = \{F = 1, G = 0, H = 1\}$. We also have: $P(E = 0 \mid A = 0, B = 1, C) = P(E = 0 \mid A = 0, B = 1)$; consequently, another such assignment is $\mathbf{x} = \{E = 0\}, \mathbf{z}_\dagger = \{A = 0, B = 1\}, \mathbf{e}_\dagger = \{H = 1\}, \mathbf{Z}_\ddagger = \{C\}, \mathbf{e}_\ddagger = \{F = 1, G = 0\}$.*

We aim to treat the evidence $\mathbf{e}_\ddagger$ independently, thus, we define it first.

Definition 6. *The diagnostic evidence $\mathbf{e}_\ddagger$ in a contextual assignment $\mathbf{x}$, $\mathbf{z}_\dagger$, $\mathbf{Z}_\ddagger$, $\mathbf{e}_\dagger$, $\mathbf{e}_\ddagger$ will be called residual evidence.*

However, contextual assignments do not immediately allow us to treat the residual evidence independently. We need the assignments to be safe.

Definition 7. *Let $e \in \mathbf{e}_\star$ be a diagnostic evidence, and let S be an unobserved ancestor of E in the graph structure in $\mathcal{B}_\star$, where $\mathcal{B}_\star$ is the sub-network formed by the requisite variables. Let $S \to \cdots B_i \cdots \to E$ be a causal trail such that either no B_i is observed or there is no B_i. Let $\mathbf{S}$ be a set of all such S. Then the variables $\mathbf{S}$ will be called basis of e. Let $\dot{\mathbf{e}}_\star \subseteq \mathbf{e}_\star$, and let $\dot{\mathbf{S}}_\star$ be a set of all such S for all $e \in \dot{\mathbf{e}}_\star$. Then $\dot{\mathbf{S}}_\star$ will be called basis of $\dot{\mathbf{e}}_\star$.*

Reconsider Example 3; the basis of $\{F = 1\}$ is $\{B\}$.

Definition 8. *Let $\mathbf{x}$, $\mathbf{z}_\dagger$, $\mathbf{Z}_\ddagger$, $\mathbf{e}_\dagger$, $\mathbf{e}_\ddagger$ be a contextual assignment, and let $\mathbf{S}_\ddagger$ be the basis of the residual evidence $\mathbf{e}_\ddagger$. If $\mathbf{S}_\ddagger \subseteq \mathbf{Z}_\ddagger$ then the contextual assignment will be called safe.*

Example 4. *Reconsider Example 3; the first example of contextual assignment is safe, but the second is not since the basis B of $\mathbf{e}_\ddagger$ is assigned in $\mathbf{z}_\dagger$. We can make the second safe like this: $\mathbf{x} = \{E = 0\}, \mathbf{z}_\dagger = \{A = 0, B = 1\}, \mathbf{e}_\dagger = \{F = 1, H = 1\}, \mathbf{Z}_\ddagger = \{C\}, \mathbf{e}_\ddagger = \{G = 0\}$. See Figure 3.*

Before showing that the residual evidence can now be treated independently, we first define a random variable called *weight*.

Definition 9. *Let $e \in \mathbf{e}_\star$ be a diagnostic evidence, and let W_e be a random variable defined as follows:*

$$W_e = P(e \mid \mathbf{Pa}(E)).$$

The variable W_e will be called weight of e. The weight of a subset $\dot{\mathbf{e}}_\star \subseteq \mathbf{e}_\star$ is defined as follows:

$$W_{\dot{\mathbf{e}}_\star} = \prod_{u_i \in \dot{\mathbf{e}}_\star} P(u_i \mid \mathbf{Pa}(U_i)).$$

Now we can show the following result:

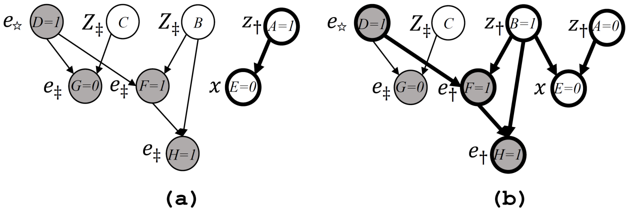


Figure 3: Two safe contextual assignments to variables of BN in Figure 1: (a) in the context $A = 1$, where edges $C \to E$ and $B \to E$ are redundant since $E \perp B, C \mid A = 1$; (b) in the context $A = 0, B = 1$, where the edge $C \to E$ is redundant since $E \perp C \mid A = 0, B = 1$. To identify such assignments, intuitively, we should apply the Bayes-ball algorithm after removing these edges. Portions of graphs that the algorithm visits, starting with visiting the variable E from its child, are highlighted. Notice that variables $\mathbf{x}, \mathbf{z}_\dagger, \mathbf{e}_\dagger$ lie in the highlighted portion.

Theorem 2. *Let $\dot{\mathbf{e}}_\star \subseteq \mathbf{e}_\star$, and let $\dot{\mathbf{S}}_\star$ be the basis of $\dot{\mathbf{e}}_\star$. Then the expectation of weight $W_{\dot{\mathbf{e}}_\star}$ relative to the distribution $Q_\star$ as defined in Equation 4 can be written as:*

$$\mathbb{E}_{Q_\star}[W_{\dot{\mathbf{e}}_\star}] = \sum_{\dot{\mathbf{s}}_\star} \prod_{u_i \in \dot{\mathbf{e}}_\star \cup \dot{\mathbf{s}}_\star} P(u_i \mid \mathbf{pa}(U_i)).$$

Hence, apart from unobserved variables $\dot{\mathbf{S}}_\star$, the computation of $\mathbb{E}_{Q_\star}[W_{\dot{\mathbf{e}}_\star}]$ does not depend on other unobserved variables. Now we are ready to show our main result:

Theorem 3. *Let Ψ be a set of all possible safe contextual assignments in the distribution P. Then the query μ to P can be computed as follows:*

$$\frac{\sum_{\psi \in \Psi} \Big(\prod_{u_i \in \mathbf{x}[\psi] \cup \mathbf{z}_\dagger[\psi] \cup \mathbf{e}_\dagger[\psi]} P(u_i \mid \mathbf{ppa}(U_i)) f(\mathbf{x}[\psi]) R[\psi] \Big)}{\sum_{\psi \in \Psi} \Big(\prod_{u_i \in \mathbf{x}[\psi] \cup \mathbf{z}_\dagger[\psi] \cup \mathbf{e}_\dagger[\psi]} P(u_i \mid \mathbf{ppa}(U_i)) R[\psi] \Big)} \tag{6}$$

where $R[\psi]$ denotes $\mathbb{E}_{Q_\star}[W_{\mathbf{e}_\ddagger[\psi]}]$.

We draw some important conclusions: i) μ can be exactly computed by performing the summation over all safe contextual assignments; notably, variables in $\mathbf{Z}_\dagger$ vary, and so does variables in $\mathbf{E}_\dagger$; ii) For all $\psi \in \Psi$, the computation of $\mathbb{E}_{Q_\star}[W_{\mathbf{e}_\ddagger[\psi]}]$ does not depend on the context $\mathbf{x}[\psi], \mathbf{z}_\dagger[\psi]$ since no basis of $\mathbf{e}_\ddagger[\psi]$ is assigned in the context (by Theorem 2). Hence, $\mathbb{E}_{Q_\star}[W_{\mathbf{e}_\ddagger[\psi]}]$ can be computed independently. However, the context decides which evidence should be in the subset $\mathbf{e}_\ddagger[\psi]$. That is why we can not cancel $\mathbb{E}_{Q_\star}[W_{\mathbf{e}_\ddagger[\psi]}]$ from the numerator and denominator.

4.2 Context-Specific Likelihood Weighting

First, we present an algorithm that simulates the DC($\mathcal{B}$) program $\mathbb{P}$, specifying the same distribution P, to generate safe contextual assignments. Then we discuss how to estimate the expectations independently before estimating μ.

Algorithm 1 Simulation of DC($\mathcal{B}$) Programs

```
procedure SIMULATE-DC($\mathbb{P}$, $\mathbf{x}$, $\mathbf{e}$)
  • Visits variables from parent and also simulates a DC($\mathcal{B}$)
    program $\mathbb{P}$ based on inputs: i) $\mathbf{x}$, a query; ii) $\mathbf{e}$: evidence.
  • Output: i) i: $f(\mathbf{x})$ that can be either 0 or 1; ii) W: a table
    of weights of diagnostic evidence ($\mathbf{e}_\dagger$).
  • The procedure maintains global data structures: i) Asg,
    a table that records assignments of variables ($\mathbf{x} \cup \mathbf{z}_\dagger$); ii)
    Dst, a table that records distributions for variables; iii)
    Forward, a set of variables whose children to be visited
    from parent; iv) Top, a set of variables marked on top; v)
    Bottom, a set of variables marked on bottom.
  1. Empty Asg, Dst, W, Top, Bottom, Forward.
  2. If PROVE-MARKED($\mathbf{x}$)==yes then i = 1 else i = 0.
  3. While Forward is not empty:
     (a) Remove m from Forward.
     (b) For all h ~ $\mathcal{D}$ ← Body in $\mathbb{P}$ such that m=z in Body:
         i. If h is observed in $\mathbf{e}$ and h not in Top:
            A. Add h to Top
            B. For all h ~ $\mathcal{D}$ ← Body in $\mathbb{P}$: PROVE-
               MARKED(Body ∧ dist(h,$\mathcal{D}$)).
            C. Let x be a observed value of h and let p be a prob-
               ability at x according to distribution Dst[h].
               Record W[h]=p.
        ii. If h is not observed in $\mathbf{e}$ and h not in Bottom:
            A. Add h to Bottom and add h to Forward.
  4. Return [i,W].
```

Algorithm 2 DC($\mathcal{B}$) Proof Procedure

```
procedure PROVE-MARKED(Goal)
  • Visits variables from child, consequently, proves a con-
    junction of atoms Goal. Returns yes; otherwise fails.
  • Accesses the program $\mathbb{P}$, the set Top, the tables Asg, Dst
    and evidence $\mathbf{e}$ as defined in Algorithm 1.
  1. While Goal in not empty:
     (a) Select the first atom b from Goal.
     (b) If b is of the form a=x:
          i. If a is observed in $\mathbf{e}$ then let y is the value of a.
         ii. Else if a in Top then y=Asg[a].
        iii. Else:
             A. Add a to Top.
             B. For all a ~ $\mathcal{D}$ ← Body in $\mathbb{P}$: PROVE-
                MARKED(Body ∧ dist(a,$\mathcal{D}$))
             C. Sample a value y from distribution Dst[a] and
                record Asg[a]=y.
             D. If a not in Bottom: add a to Bottom and add
                a to Forward.
         iv. If x==y then remove b from Goal else fail.
     (c) If b is of the form dist(a,$\mathcal{D}$): record Dst[a]=$\mathcal{D}$
         and remove b from Goal.
  2. Return yes.
```

4.2.1 Simulation of DC($\mathcal{B}$) Programs

We start by asking a question. Suppose we modify the first and the fourth rule of Bayes-ball simulation, introduced in Section 3, as follows:

- In the first rule, when the visit of an unobserved variable is from its child, everything remains the same except that only some parents are visited, not all.
- Similarly, in the fourth rule, when the visit of an observed variable is from its parent, everything remains the same except that only some parents are visited.

Which variables will be assigned, and which will be weighted using the modified simulation rules? Intuitively, only a subset of variables in $\mathbf{Z}_\star$ should be assigned, and only a subset of variables in $\mathbf{E}_\star$ should be weighted. But then how to assign/weigh a variable knowing the state of only some of its parent. We can do that when structures are present in CPDs, and these structures are explicitly represented using rules, as discussed in Section 2.3. This is because rules define the distribution from which the variable should be sampled, although the state of some parents of the variable is known before that. Hence, the key idea is to visit only some parents (if possible due to structures); consequently, those unobserved parents that are not visited might not be required to be sampled.

To realize that, we need to modify the Bayes-ball simulation such that it works on DC($\mathcal{B}$) programs. This modified simulation for DC($\mathcal{B}$) programs is defined procedurally in Algorithm 1. The algorithm visits variables from their parents and calls Algorithm 2 to visit variables from their children. Like Bayes-ball, this algorithm also marks variables on top and bottom to avoid repeating the same action. Readers familiar with theorem proving will find that Algorithm 2 closely resembles SLD resolution (Kowalski, 1974), but it is also different since it is stochastic. An example illustrating how Algorithm 2 visits only some requisite ancestors to sample a variable is present in the supplementary material.

Since the simulation of $\mathbb{P}$ follows the same four rules of Bayes-ball simulation except that only some parents are visited in the first and fourth rule, we show that

Lemma 4. *Let $\mathbf{E}_\dagger$ be a set of observed variables weighed and let $\mathbf{Z}_\dagger$ be a set of unobserved variables, apart from query variables, assigned in a simulation of $\mathbb{P}$, then,*

$$\mathbf{Z}_\dagger \subseteq \mathbf{Z}_\star \text{ and } \mathbf{E}_\dagger \subseteq \mathbf{E}_\star.$$

The query variables $\mathbf{X}$ are always assigned since the simulation starts with visiting these variables as if visits are from one of their children. To simplify notation, from now on we use $\mathbf{Z}_\dagger$ to denote the subset of variables in $\mathbf{Z}_\star$ that are assigned, $\mathbf{E}_\dagger$ to denote the subset of variables in $\mathbf{E}_\star$ that are weighted in the simulation of $\mathbb{P}$. $\mathbf{Z}_\ddagger$ to denote $\mathbf{Z}_\star \setminus \mathbf{Z}_\dagger$, and $\mathbf{E}_\ddagger$ to denote $\mathbf{E}_\star \setminus \mathbf{E}_\dagger$ We show that the simulation performs safe contextual assignments to requisite variables.

Theorem 5. *The partial assignment $\mathbf{x}$, $\mathbf{z}_\dagger$, $\mathbf{Z}_\ddagger$, $\mathbf{e}_\dagger$, $\mathbf{e}_\ddagger$ generated in a simulation of $\mathbb{P}$ is a safe contextual assignment.*

The proof of Theorem 5 relies on the following Lemma.

Lemma 6. *Let $\mathbb{P}$ be a DC($\mathcal{B}$) program specifying a distribution P. Let $\mathbf{B}, \mathbf{C}$ be disjoint sets of parents of a variable A. In the simulation of $\mathbb{P}$, if A is sampled/weighted, given an assignment $\mathbf{c}$, and without assigning $\mathbf{B}$, then,*

$$P(A \mid \mathbf{c}, \mathbf{B}) = P(A \mid \mathbf{c}).$$

Hence, just like the standard LW, we sample from a factor $Q_\dagger$ of the proposal distribution $Q_\star$, which is given by,

$$Q_\dagger = \prod_{u_i \in \mathbf{x} \cup \mathbf{z}_\dagger \cup \mathbf{e}_\dagger} P(u_i \mid \mathbf{ppa}(U_i))$$

where $P(u_i \mid \mathbf{ppa}(U_i)) = 1$ if $u_i \in \mathbf{e}_\dagger$. It is precisely this factor that Algorithm 1 considers for the simulation of $\mathbb{P}$. Starting by first setting $\mathbf{E}_\star$, $\mathbf{E}_\ddagger$ their observed values, it assigns $\mathbf{X} \cup \mathbf{Z}_\dagger$ and weighs $\mathbf{e}_\dagger$ in the topological ordering. In this process, it records *partial weights* $\mathbf{w}_{\mathbf{e}_\dagger}$, such that: $\prod_{x_i \in \mathbf{e}_\dagger} w_{x_i} = w_{\mathbf{e}_\dagger}$ and $w_{x_i} \in \mathbf{w}_{\mathbf{e}_\dagger}$. Given M partially weighted samples $\mathcal{D}_\dagger = \langle \mathbf{x}[1], \mathbf{w}_{\mathbf{e}_\dagger[1]} \rangle, \dots, \langle \mathbf{x}[M], \mathbf{w}_{\mathbf{e}_\dagger[M]} \rangle$ from $Q_\dagger$, we could estimate μ using Theorem 3 as follows:

$$\overline{\mu} = \frac{\sum_{m=1}^{M} f(\mathbf{x}[m]) \times w_{\mathbf{e}_\dagger[m]} \times \mathbb{E}_{Q_\star}[W_{\mathbf{e}_\ddagger[m]}]}{\sum_{m=1}^{M} w_{\mathbf{e}_\dagger[m]} \times \mathbb{E}_{Q_\star}[W_{\mathbf{e}_\ddagger[m]}]} \tag{7}$$

However, we still can not estimate it since we still do not have expectations $\mathbb{E}_{Q_\star}[W_{\mathbf{e}_\ddagger[m]}]$. Fortunately, there are ways to estimate them from partial weights in $\mathcal{D}_\dagger$. We discuss one such way next.

4.2.2 Estimating the Expected Weight of Residuals

We start with the notion of sampling mean. Let $\mathcal{W}_\star = \langle w_{e_1}[1], \dots, w_{e_m}[1] \rangle, \dots, \langle w_{e_1}[n], \dots, w_{e_m}[n] \rangle$ be a data set of n observations of weights of m diagnostic evidence drawn using the standard LW. How can we estimate the expectation $\mathbb{E}_{Q_\star}[W_{e_i}]$ from $\mathcal{W}_\star$? The standard approach is to use the sampling mean: $\overline{W}_{e_i} = \frac{1}{n} \sum_{r=1}^{n} w_{e_i}[r]$. In general, $\mathbb{E}_{Q_\star}[W_{e_i} \dots W_{e_j}]$ can be estimated using the estimator: $\overline{W_{e_i} \dots W}_{e_j} = \frac{1}{n} \sum_{r=1}^{n} w_{e_i}[r] \dots w_{e_j}[r]$. Since LW draws are independent and identical distributed (i.i.d.), it is easy to show that the estimator is unbiased.

However, some entries, i.e., weights of residual evidence, are missing in the data set $\mathcal{W}_\dagger$ obtained using CS-LW. The trick is to fill the missing entries by drawing samples of the missing weights once we obtain $\mathcal{W}_\dagger$. More precisely, missing weights $\langle W_{e_i}, \dots, W_{e_j} \rangle$ in r^{th} row of $\mathcal{W}_\dagger$ are filled in with a joint state $\langle w_{e_i}[r], \dots, w_{e_j}[r] \rangle$ of the weights. To draw the joint state, we again use Algorithm 1 and visit observed variables $\langle E_i, \dots, E_j \rangle$ from parent. Once all missing entries are filled in, we can estimate $\mathbb{E}_{Q_\star}[W_{e_i} \dots W_{e_j}]$ using the estimator $\overline{W_{e_i} \dots W}_{e_j}$ as just discussed. Once we estimate all required expectations, it is straightforward to estimate μ using Equation 7.

		LW		CS-LW	
BN	**N**	**MAE ± Std.**	**Time**	**MAE ± Std.**	**Time**
Alarm	100	0.2105 ± 0.1372	0.09	0.0721 ± 0.0983	0.06
	1000	0.0766 ± 0.0608	0.86	0.0240 ± 0.0182	0.53
	10000	0.0282 ± 0.0181	8.64	0.0091 ± 0.0069	5.53
	100000	0.0086 ± 0.0067	89.93	0.0034 ± 0.0027	57.64
Andes	100	0.0821 ± 0.0477	1.07	0.0619 ± 0.0453	0.22
	1000	0.0257 ± 0.0184	10.62	0.0163 ± 0.0139	2.20
	10000	0.0087 ± 0.0069	106.55	0.0058 ± 0.0042	22.62
	100000	0.0025 ± 0.0015	1074.93	0.0020 ± 0.0016	233.72

Table 1: The mean absolute error (MAE), the standard deviation of the error (Std.), and the average elapsed time (in seconds) versus the number of samples (N). For each case, LW and CS-LW were executed 30 times.

At this point, we can gain some insight into the role of CSIs in sampling. They allow us to estimate the expectation $\mathbb{E}_{Q_\star}[W_{\mathbf{e}_\ddagger}]$ separately. We estimate it from all samples obtained at the end of the sampling process, thereby reducing the contribution $W_{\mathbf{e}_\ddagger}$ makes to the variance of our main estimator $\overline{\mu}$. The residual evidence $\mathbf{e}_\ddagger$ would be large if much CSIs are present in the distribution; consequently, we would obtain a much better estimate of μ using significantly fewer samples. Moreover, drawing a single sample would be faster since only a subset of requisite variables is visited. Hence, in addition to CIs, we exploit CSIs and improve LW further. We observe all these speculated improvements in our experiments.

5 Empirical Evaluation

We answer three questions empirically:

Q1: How does the sampling speed of CS-LW compare with the standard LW in the presence of CSIs?

Q2: How does the accuracy of the estimate obtained using CS-LW compare with the standard LW ?

Q3: How does CS-LW compare to the state-of-the-art approximate inference algorithms?

To answer the first two questions, we need BNs with structures present within CPDs. Such BNs, however, are not readily available since the structure while designing inference algorithms is generally overlooked. We identified two BNs from the Bayesian network repository (Elidan, 2001), which have many structures within CPDs: i) *Alarm*, a monitoring system for patients with 37 variables; ii) *Andes*, an intelligent tutoring system with 223 variables.

We used the standard decision tree learning algorithm to detect structures and overfitted it on tabular-CPDs to get tree-CPDs, which was then converted into rules. Let us denote the program with these rules by $\mathbb{P}_{tree}$. CS-LW is implemented in the Prolog programming language, thus to compare the sampling speed of LW with CS-LW, we need a similar implementation of LW. Fortunately, we can use the same implementation of CS-LW for obtaining LW estimates. Recall that if we do not make structures explicit in rules and represent each entry in tabular-CPDs with rules, then CS-LW boils down to LW. Let $\mathbb{P}_{table}$ denotes the program where each rule in it corresponds to an entry in tabular-CPDs. Table 1 shows the comparison of estimates obtained using $\mathbb{P}_{tree}$ (CS-LW) and $\mathbb{P}_{table}$ (LW). Note that CS-LW automatically discards non-requisite variables for sampling. So, we chose the query and evidence such that almost all variables in BNs were requisite for the conditional query.

As expected, we observe that less time is required by CS-LW to generate the same number of samples. This is because it visits only the subset of requisite variables in each simulation. *Andes* has more structures compared to *Alarm*. Thus, the sampling speed of CS-LW is much faster compared to LW in *Andes*. Additionally, we observe that the estimate, with the same number of samples, obtained by CS-LW is much better than LW. This is significant. It is worth emphasizing that approaches based on collapsed sampling obtain better estimates than LW with the same number of samples, but then the speed of drawing samples significantly decreases. In CS-LW, the speed increases when structures are present. This is possible because CS-LW exploits CSIs.

Hence, we get the answer to the first two questions: When many structures are present, and when they are made explicit in rules, then CS-LW will draw samples faster compared to LW. Additionally, estimates will be better with the same number of samples.

To answer our final question, we compared CS-LW with the collapsed compilation (CC, Friedman and Van den Broeck, 2018), which has been recently shown to outperform several sampling algorithms. It combines a state of the art exact inference algorithm that exploits CSIs and importance sampling that scales the exact inference. The load between the exact and sampling can be regulated using the size of the arithmetic circuit: larger the circuit's size, larger the load on the exact and lesser the load on the sampling, i.e., less variables are considered for sampling. For this experiment, we consider two additional BNs: i) *Win95pts*, a system for printing troubleshooting in Windows 95 with 76 variables; ii) *Munin1*, an expert EMG assistant with 186 variables. However, not many structures are present in the CPDs of these two BNs, so not much difference in the performance of LW and CS-LW is expected.

The comparison is shown in Table 2. We can observe the following: i) as expected from collapsed sampling, much fewer samples are drawn in the same time; ii) the right choice of circuit's size is crucial, e.g., with circuit size 10,000, CC performs poorly compared to LW on some BNs while better when the size is increased; iii) CS-LW performs better compared to CC when the circuit is not huge; iv) on the three BNs, CC with a huge circuit size computes the exact conditional probability while LW and CS-LW can only provide a good approximation of that in the same time.

To demonstrate that the fourth observation does not undermine the importance of pure sampling, we used *Munin1*. Although the size of this BN is comparable to the size of *Andes*, almost all variables are multi-valued, and their domain size can be as large as 20; hence, some CPDs are huge, while in *Andes*, variables are binary-valued. CC that works well on *Andes*, fails to deal with huge CPDs of *Munin1* on a machine with 16 GB memory. On the other hand, both LW and CS-LW work well on this BN.

Hence, we get the answer to our final question: CS-LW is competitive with the state-of-the-art and can be a useful tool for inference on massive BNs with structured CPDs.

6 Related Work

Although the question of how to exploit CSIs arising due to structures is not new and has puzzled researchers for decades, research in this direction has mainly been focused on exact inference (Boutilier et al., 1996; Zhang and Poole, 1999; Poole, 1997; Poole and Zhang, 2003). Nowadays, it is common to use knowledge compilation (KC) based exact inference for the purpose (Chavira and Darwiche, 2008; Fierens et al., 2015; Shen et al., 2016). There are not many approximate inference algorithms that can exploit them. However, there are some tricks that make use of structures to approximate the probability of a query. One trick, introduced by Poole (1998), is to make rule-base simpler by ignoring distinctions in close probabilities. Another trick, explored in the case of the tree-CPDs, is to prune trees and reduce the size of actual CPDs (Salmerón et al., 2000; Cano et al., 2011). However, approximation by making distribution simpler is orthogonal to traditional ways of approximation, such as sampling. Fierens (2010) observed the speedup in Gibbs sampling due to structures; however, did not consider global implications of structures.

Recently, Friedman and Van den Broeck (2018) realized that KC is good at exploiting structures while sampling is scalable; thus, proposed CC that inherits advantages of both. However, along with advantages, this approach also inherits the scalability limitations of KC. Furthermore, CC is limited to discrete distributions. The problem of exploiting CSIs in discrete-continuous distributions is non-trivial and is poorly studied. Recently, it has attracted some attention (Zeng and Van den Broeck, 2019). However, proposed approaches are also exact and rely on complicated weighted model integration (Belle et al., 2015), which quickly become infeasible. CS-LW is simple, scalable, and applies to such distributions. A sampling algorithm for a rule-based representation of discrete-continuous distributions was developed by Nitti et al. (2016); however, it did not exploit CIs and global implications of rule structures.

	LW		**CC-10,000**		**CC-100,000**		**CC-1000,000**		**CS-LW**	
BN	**N**	**MAE ± Std.**	**N**	**MAE ± Std.**	**N**	**MAE ± Std.**	**N**	**MAE ± Std.**	**N**	**MAE ± Std.**
Alarm	131606	0.0073 ± 0.0054	3265	0.0022 ± 0.0018	NA	0 ± 0 (exact)	NA	0 ± 0 (exact)	178620	0.0019 ± 0.0016
Win95pts	51956	0.0022 ± 0.0016	635	0.0149 ± 0.0163	NA	0 ± 0 (exact)	NA	0 ± 0 (exact)	67855	0.0017 ± 0.0011
Andes	13113	0.0068 ± 0.0062	116	0.0814 ± 0.0915	17	0.0060 ± 0.0094	NA	0 ± 0 (exact)	56672	0.0022 ± 0.0021
Munin1	15814	0.0036 ± 0.0026	out of memory		out of memory		out of memory		17985	0.0035 ± 0.0025

Table 2: The mean absolute error (MAE), the standard deviation of the error (Std.), and the average number of samples (N) drawn when algorithms were run 50 times for 2 minutes (approx.) each. The algorithms are: LW, CS-LW, CC with circuit size 10,000, with size 100,000, and with size 1000,000.

7 Conclusion

We studied the role of CSI in approximate inference and introduced a notion of contextual assignments to show that CSIs allow for breaking the main problem of estimating conditional probability query into several small problems that can be estimated independently. Based on this notion, we presented an extension of LW, which not only generates samples faster; it also provides a better estimate of the query with much fewer samples. Hence, we provided a solid reason to use structured-CPDs over tabular-CPDs. Like LW, we believe other sampling algorithms can also be extended along the same line. We aim to open up a new direction towards improved sampling algorithms that also exploit CSIs.

Acknowledgements This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No [694980] SYNTH: Synthesising Inductive Data Models). OK was supported by the Czech Science Foundation project "Generative Relational Models" (20-19104Y) and partially also by the OP VVV project *CZ.02.1.01/0.0/0.0/16_019/0000765* "Research Center for Informatics". The authors would like to thank Luc De Raedt, Jessa Bekker, Pedro Zuidberg Dos Martires and the anonymous reviewers for valuable feedback.

References

Craig Boutilier, Nir Friedman, Moises Goldszmidt, and Daphne Koller. Context-specific independence in bayesian networks. In *Proceedings of the Twelfth international conference on Uncertainty in artificial intelligence*, pages 115–123. Morgan Kaufmann Publishers Inc., 1996.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

David Maxwell Chickering, David Heckerman, and Christopher Meek. A bayesian approach to learning bayesian networks with local structure. In *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence*, pages 80–89. Morgan Kaufmann Publishers Inc., 1997.

Nir Friedman. The bayesian structural em algorithm. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 129–138. Morgan Kaufmann Publishers Inc., 1998.

John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.

Tal Friedman and Guy Van den Broeck. Approximate knowledge compilation by online collapsed importance sampling. In *Advances in Neural Information Processing Systems*, pages 8024–8034, 2018.

Adnan Darwiche. A differential approach to inference in bayesian networks. *Journal of the ACM (JACM)*, 50(3): 280–305, 2003.

Ross D Shachter and Mark A Peot. Simulation approaches to general probabilistic inference on belief networks. In *Machine Intelligence and Pattern Recognition*, volume 10, pages 221–231. Elsevier, 1990.

Robert Fung and Kuo-Chu Chang. Weighing and integrating evidence for stochastic simulation in bayesian networks. In *Machine Intelligence and Pattern Recognition*, volume 10, pages 209–219. Elsevier, 1990.

David Poole. Probabilistic partial evaluation: Exploiting rule structure in probabilistic inference. In *IJCAI*, volume 97, pages 1284–1291, 1997.

Jukka Corander, Antti Hyttinen, Juha Kontinen, Johan Pensar, and Jouko Väänänen. A logical approach to context-specific independence. *Annals of Pure and Applied Logic*, 170(9):975–992, 2019.

Bernd Gutmann, Ingo Thon, Angelika Kimmig, Maurice Bruynooghe, and Luc De Raedt. The magic of logical inference in probabilistic programming. *Theory and Practice of Logic Programming*, 11(4-5):663–680, 2011.

R Shacter. Bayes ball: The rational pastime. In *Proc of the 14 Annual Conf on Uncertainty in Artificial Intelligence*, 1998.

Robert Kowalski. Predicate logic as programming language. In *IFIP congress*, volume 74, pages 569–544, 1974.

G. Elidan. Bayesian Network Repository, 2001. https://www.cse.huji.ac.il/ galel/Repository/.

Nevin L Zhang and David Poole. On the role of context-specific independence in probabilistic inference. In *16th International Joint Conference on Artificial Intelligence, IJCAI 1999, Stockholm, Sweden*, volume 2, page 1288, 1999.

David Poole and Nevin Lianwen Zhang. Exploiting contextual independence in probabilistic inference. *Journal of Artificial Intelligence Research*, 18:263–313, 2003.

Mark Chavira and Adnan Darwiche. On probabilistic inference by weighted model counting. *Artificial Intelligence*, 172(6-7):772–799, 2008.

Daan Fierens, Guy Van den Broeck, Joris Renkens, Dimitar Shterionov, Bernd Gutmann, Ingo Thon, Gerda Janssens, and Luc De Raedt. Inference and learning in probabilistic logic programs using weighted boolean formulas. *Theory and Practice of Logic Programming*, 15(3):358–401, 2015.

Yujia Shen, Arthur Choi, and Adnan Darwiche. Tractable operations for arithmetic circuits of probabilistic models. In *Advances in Neural Information Processing Systems*, pages 3936–3944, 2016.

David Poole. Context-specific approximation in probabilistic inference. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, page 447–454, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 155860555X.

Antonio Salmerón, Andrés Cano, and Serafın Moral. Importance sampling in bayesian networks using probability trees. *Computational Statistics & Data Analysis*, 34(4): 387–413, 2000.

Andrés Cano, Manuel Gémez-Olmedo, and Serafén Moral. Approximate inference in bayesian networks using binary probability trees. *International Journal of Approximate Reasoning*, 52(1):49–62, 2011.

Daan Fierens. Context-specific independence in directed relational probabilistic models and its influence on the efficiency of gibbs sampling. In *ECAI*, pages 243–248, 2010.

Zhe Zeng and Guy Van den Broeck. Efficient search-based weighted model integration. *UAI 2019 Proceedings*, 2019.

Vaishak Belle, Andrea Passerini, and Guy Van den Broeck. Probabilistic inference in hybrid domains by weighted model integration. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

Davide Nitti, Tinne De Laet, and Luc De Raedt. Probabilistic logic programming for hybrid relational domains. *Machine Learning*, 103(3):407–449, 2016.